Graph Slam - Factor Graph Approach to Simultaneous Localization and Mapping

Ignacio Torroba Balmori, Daniel Duberg and John Folkesson

July 2019

1 Introduction

Simultaneous localization and mapping, SLAM, is an essential requirement for mobile robots to be able to function in unstructured environments. Robots have a surprisingly difficult time understanding where they are relative to things in the space around them. Motion through that space can be estimate with the help of inertial sensors or wheel encoders but this is exactly as a person moving with a blindfold. Errors accumulate quickly and it is impossible to navigate very long in this way.

To allow more accurate motion and to put the robot into a context of other objects, some sensing of the environment is required. That can be a camera, sonar, or lidar sensor. By matching readings in two measurements it is possible to constrain the motion estimate between the robot poses from which those measurements were taken. That is the essence of SLAM. Formulating that as an algorithm is difficult due to the many constraints that must be accurately estimated and then optimized.

In this tutorial we will look at the factor graph approach to formulating this problem and try out a simple Graph Slam implementation for point landmarks in 2D measured by a lidar sensor from robot with wheel encoders. First the factor's will be discussed then a (simple) solution method. After that there are some practical exercises in Matlab that will allow one to make some modifications to the simple approach for common problems that arise.

1.1

1.1 Tools and requirements

- A Matlab skeleton of the exercises is provided, which the user shall complete in order to implement changes to the GraphSLAM algorithm introduced in the theoretical part.
- The data sets required to run the simulations has been taken from the course Applied Estimation (EL2320) and its content is explained in the practical part.

- The exercises presented touch upon both the theoretical and practical side of working with PGMs and SLAM, and therefore a basic understanding of both. The theory and implementation details can be found in [1-3] for the reader with interest in graph-based SLAM solutions.

1.2 The full SLAM problem

Formally, we want to compute the distribution in eq. (1) in times $i = 0, ..., N$ over all the robot poses $\mathbf{x}_{0:N}$ and map landmarks $\mathbf{m} = \mathbf{m}_1, ..., \mathbf{m}_M$, given a set of measurements $\mathbf{z}_{1:K}$, and controls $\mathbf{u}_{1:N}$.

$$p(\mathbf{x_{0:N}}, \mathbf{m}_{1:M} \mid \mathbf{z}_{1:K}, \mathbf{u}_{1:N}) \tag{1}$$

A few assumptions are usually made in order to work with (1), the most important one we will make here being that all the sensor and motion noise follow normal distributions. However the sensors are indirectly and nonlinearly related to the pose and map coordinates. This tutorial ia devoted to full SLAM solutions, in which all robot poses and map landmarks are computed in a batch fashion, instead of incrementally in a recursive algorithm. We will see how this has its disadvantages and investigate an intermediate approach. The control and measurement inputs generate spatial constraints between robot poses and landmarks, minimizing the error in these variables will yield the configuration of all robot poses and landmarks that best explain the inputs received. Thus, SLAM can be posed as an optimization problem.

2 SLAM as a Probabilistic Graphical Model

The structure of the SLAM problem and how states, measurements, and controls evolve over time leads to a specific structure in the factor graph that can be exploited in various ways. There are three main graphical models for representing the SLAM problem, [4].

2.1 Bayesian Belief Network

The full SLAM problem can be formulated as a Bayesian belief network (BBN). A BBN is a directed acyclic graph which encodes a set of conditional independence assumptions. The nodes in the graph are random variables and the edges represent direct dependence between two nodes. A variable only directly depends on its predecessors in the graph. If x_i is the robot state at the i^{th} time step, m_j the j^{th} landmark, and z_k the k^{th} measurement, with $i \in 0...N$, $j \in 1...M$, and $k \in 1...K$, as in [4]. The joint probability model for the BBN can be factored as:

$$p(x_{0:N}, m_{1:M}, z_{1:K}, u_{1:N}) = p(x_0) \prod_{i=1}^{N} p(x_i \mid x_{i-1}, u_i) \prod_{k=1}^{K} p(z_k | x_{p(k)}, m_{f(k)}) \tag{2}$$

where $p(x_0)$ is a prior of the initial state, $p(x_i \mid x_{i-1}, u_i)$ the motion model with u_i being the control input, and $p(z_k|x_{p(k)}, m_{f(k)})$ the landmark measurement model. The mapping function $p(k)$ and $f(k)$ map to index of the pose and feature involved in the k^{th} measurement as given by the data association. For the SLAM problem the nodes would in this case be each pose x_i, each control input u_i, each measurement z_k, each landmarks m_j .

Equation (2) leads to a recursive estimator such as an extended Kalman filter as the factors can be separated at each step as new measurements are added by multiplying the previous estimate. One can see the separate models for each measurement explicitly in the formula.

2.2 Factor Graph

By representing the SLAM problem as a factor graph, the underlying optimization problem is made more clear. In the factor graph representation the known variables from the Bayesian belief network, the measurements $\mathbf{z}_k$ and the control inputs $\mathbf{u}_i$, are eliminated. These variables are instead added as factor nodes connecting the nodes that the measurement depends on. The SLAM problem represented by a factor graph would have nodes for the robot poses $\mathbf{x}_i$ and the landmarks $\mathbf{m}_j$. There would be factor nodes between robot poses $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ that corresponds to the control input $\mathbf{u}_i$, and factor nodes between various robot poses and the landmarks that corresponds to the measurements, $\mathbf{z}_k$.

2.3 Markov Random Field

Lastly, the SLAM problem can be represented by Markov random fields (MRFs). In the MRF graph the factor nodes have been eliminated and the graph is undirected. This means that the only nodes in the graph are the robot poses x 0:M and the landmarks m 1:N . The edges in the graph express which variables are linked by a common factor.

2.4 Questions/exercises

1. Draw a figure of the SLAM problem as a Bayesian belief network, factor graph, and Markov random field with at least four robot poses and three landmarks.

2. What assumptions are made in Equation (2)?

3 GraphSLAM Graph construction

The GraphSLAM algorithm presented in this tutorial solves the offline SLAM problem. The first part is to construct a graph given a set of controls $\mathbf{u}_{1:N}$, and measurements $\mathbf{z}_{1:K}$ with associated correspondence variables $c_{1:K}$. The correspondence variables are the index of the landmark

being measured by the particular $\mathbf{z}_k$. This is known as the front-end in GraphSLAM.

The type of graph that is constructed is a factor graph. The nodes in the graph are therefore the robot poses,$\mathbf{x_{0:N}}$, and the landmarks, $\mathbf{m}_{1:M}$. The edges between nodes represent soft constraints and imply a factor involving the two attached nodes. An edge between two robot poses corresponds to a motion event and the edges between a robot pose and landmarks correspond to a measurement. Alternatively, one could show the factors explicitly as factor nodes as in a bipartite graph. There are typically no triplet factor nodes in the SLAM graph so this convention works in this case.

There can be, however, singleton factors, an edge terminated on only one end. These arise from measurements of absolute position such as GPS. Without any such measurements the absolute position and orientation of the solution is not observable. Since one only has relative measurements between nodes one can transform the solution with translations an rotation. To avoid this one often anchors the start position with a strong singleton factor to, for example, the origin. One then keeps in mind that the entire map can be transformed to another start pose.

A factor is often chosen to be given by a Gaussian distribution over the relative pose between two pose nodes:

$$N(\mathbf{x}_i - \mathbf{g}(\mathbf{x}_{i-1}, \mathbf{u}_i), R), \tag{3}$$

or a function of the relative position of a landmark to a pose.

$$N(\mathbf{h}(\mathbf{m}_j, \mathbf{x}_i) - \mathbf{z}_k, Q), \tag{4}$$

This system is often solved as a maximum likelihood estimation problem. It is easiest then to work with the negative log likelihood which apart from the constant normalization terms is a sum of two types of terms:

$$\frac{1}{2}(\mathbf{x}_i - \mathbf{g}(\mathbf{x}_{i-1}, \mathbf{u}_i))^T R^{-1}(\mathbf{x}_i - \mathbf{g}(\mathbf{x}_{i-1}, \mathbf{u}_i)) \tag{5}$$

and

$$\frac{1}{2}(\mathbf{h}(\mathbf{m}_j, \mathbf{x}_i) - \mathbf{z}_k)^T Q^{-1}(\mathbf{h}(\mathbf{m}_j, \mathbf{x}_i) - \mathbf{z}_k) \tag{6}$$

For a linear system the factor constraints are collected to entries in an information matrix Ω and an information vector $\boldsymbol{\xi}$:

$$\frac{1}{2}((\mathbf{x}_0; ..\mathbf{x}_N; \mathbf{m}_1; ..\mathbf{m}_M) - \Omega^{-1}\boldsymbol{\xi})^T \Omega((\mathbf{x}_0; ..\mathbf{x}_N; \mathbf{m}_1; ..\mathbf{m}_M) - \Omega^{-1}\boldsymbol{\xi}) \tag{7}$$

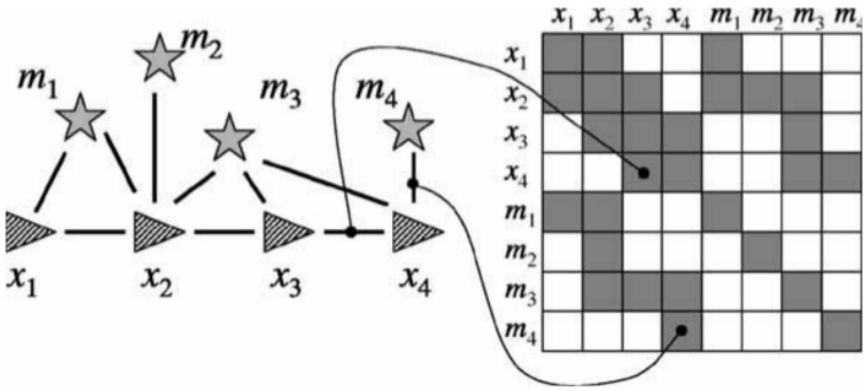

Figure 1: GraphSLAM: adding new measurements to the graph. Note that the Ω matrix is symetric and that each measurement will add to the diagonal as well as the symmetric off diagonal blocks.

Where we mean that $(\mathbf{x}_0; ..\mathbf{x}_N; \mathbf{m}_1; ..\mathbf{m}_M)$ is actually a column vector formed by stacking all the individual vectors. Leading to the simple minimization condition:

$$\Omega(\mathbf{x}_0; ..\mathbf{x}_N; \mathbf{m}_1; ..\mathbf{m}_M) = \boldsymbol{\xi} \tag{8}$$

When dealing with non-linear systems we can linearize around an estimated solution to form the Ω matrix and the $\boldsymbol{\xi}$ vector representation of the graph.

To the left in Figure (1) a factor graph can be seen, with robot poses represented by triangles and landmarks by stars. The corresponding information matrix, Ω, can be seen to the right. The gray cells in the information matrix indicate non-zero blocks due to a constraint (an edge in the graph) between either two robot poses or a robot pose and a landmark.

To incorporate a measurement $\mathbf{z}_7$ into the graph we first acknowledge that a measurement gives information between the location of the robot pose $\mathbf{x_4}$ and the landmark $\mathbf{m}_4$. This means that an edge, or constraint, between the robot pose and the landmark should be added. Practically this is done by updating the entries in Ω and $\boldsymbol{\xi}$.

The way to incorporate a control is similar. The function g is the kinematic motion model and R the motion noise covariance. The graph after all controls and measurements have been incorporated is sparse, since the number of constraints in the graph is linear in time but the size of the matrix grows quadratically. The resulting information matrix Ω has mostly zeros for the off-diagonal elements. The only off- diagonal elements that are not zero are either between a pose and a landmark if that landmark was observed from that pose, or between two consecutive poses, as can

be seen in Figure 1. Notice crucially that the entire bottom right part between the landmarks is block diagonal with no connections between landmarks. We will exploit this in solving the system.

3.1 Graph inference

A directed graph of the SLAM problem is just a representation of soft spatial constraints between robot poses and map features. Solving the graph then consists in finding the configuration of the robot poses and landmarks that best explains the observations encoded in this graph. A straight ahead solution would look like:

$$\Sigma = \Omega^{-1}, \qquad \mu = \Sigma\boldsymbol{\xi} \tag{9}$$

Where μ is the MLE solution for the full state vector. Solving this system can be achieved in linear time if each map feature has been seen only locally in time. If that is the case, the variables in Ω can be reordered to make it band-diagonal and thus it becomes simple to solve.

3.2 Questions/exercises

3. What happens to the structure of Ω when the robot sees a previously seen landmark after a long time (aka loop closure)? Hint, take a look at Figure (1). Can (9) be solved in linear time then?

4. Show that linearizing the motion equation for one pose to pose factor gives rise to the following terms to be added to Ω and $\boldsymbol{\xi}$ at the appropriate places:

$$\begin{array}{ll} \Delta\Omega_{i-1,i-1} & = G^T R^{-1} G \\ \Delta\Omega_{i,i} & =? \\ \Delta\Omega_{i,i-1} & =? \\ \Delta\boldsymbol{\xi}_{i-1} & = G^T R^{-1}(G\bar{\mu}_{i-1} - \mathbf{g}(\bar{\mu}_{i-1}, u_i) \\ \Delta\boldsymbol{\xi}_i & =? \end{array}$$

What are G and $\bar{\mu}_{i-1}$? Hint: Consider adding a new term to the log likelihood and then compare linear and quadratic terms with the form of Eq. (7).

3.3 Reducing the graph

Solving Eq. (9) becomes an expensive operation for big robot trajectories and the presence of cycles in the graph. We can at least easily eliminate map variables from Eq. (8). If we split Ω into four parts;

$$\begin{pmatrix} \Omega_{x,x} & \Omega_{x,m} \\ \Omega_{m,x} & \Omega_{m,m} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{m} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\xi_x} \\ \boldsymbol{\xi_m} \end{pmatrix}$$

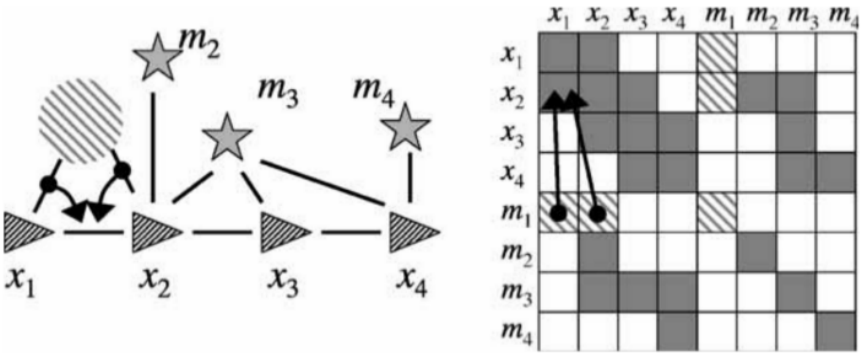

Figure 2: We show how one can reduce the size of by eliminating landmark varaibles.

We see that the reduce, pose only, system will have

$$\hat{\Omega} = \Omega_{x,x} - \Omega_{x,m}\Omega_{m,m}^{-1}\Omega_{m,x} \tag{10}$$

$$\hat{\boldsymbol{\xi}} = \boldsymbol{\xi}_x - \Omega_{x,m}\Omega_{m,m}^{-1}\boldsymbol{\xi}_m \tag{11}$$

Notice that inverting $\Omega_{m,m}$ is made easy as it is block diagonal. This in fact allow one to work up from the last landmark one landmark at a time and as each landmark is only seen from some of the poses this is linear complexity in the number of landmarks. Fig. (2) shows how this works. In general this will create more fill in the $\hat{\Omega}$ intrducing terms between pose nodes that the landmark was observed from. Previously the $\Omega_{x,x}$ only had terms between adjacent pose nodes. It is these terms of course that lead to improved estimation of the motion but they can make solving the system harder. One could do this the other way round and eliminate the pose nodes first leading to a dense map system. The $\Omega_{x,x}$ is not block diagonal but it is band diagonal and can be inverted with linear complexity also.

There are other methods for solving this and generally one tries to find an optimal ordering of the elimination to not cause too much fill in the remaining system. Clever schemes can achieve $NLogN$ complexity. Another more important issue is numerical stability. This information form is not good for that. One may use a square root form which leads to much better conditioned matrices. Generally, when the ratio of the biggest to smallest eigen value is round 10^{15} problems will start. By taking the square root the ratio is reduced to $10^{7.5}$ which will not cause issues.

3.4 Solving the system

We can solve for the robot poses by,

$$\boldsymbol{\mu}_x = \hat{\Omega}^{-1}\hat{\boldsymbol{\xi}}$$

and for the map by

$$\boldsymbol{\mu}_m = \Omega_{m,m}^{-1}(\boldsymbol{\xi}_m + \Omega_{m,x}\hat{\boldsymbol{\xi}})$$

3.5 Iterative solution

So far we have set up all the necessary steps to initialize, linearize, reduce and solve the graph. However, given the non-linearity in our sensor models, finding the graph configuration that minimizes the overall error requires us to iterate the procedure until no substantial changes are made. Hence, the last step towards implementing the GraphSLAM algorithm consists in running the linearization, reduction, and solution steps with the robot configuration resulting from the previous iteration as input for the next one.

3.6 Questions/exercises

5. How can we set a convergence condition to stop the algorithm?
6. What are the factors that might influence the number of iterations required by the algorithm?
7. When might this implementation of a GraphSLAM solution fail to converge?

4 Practical

In this section we are going to implement the GraphSLAM algorithm introduced in [2]. You have been provided the code, The data provided consists of a simulation of a mobile robot moving in 2 dimensions with wheel encoders and range-bearing sensor measurements of landmarks on a map. At very time step t the data set contains:

- Odometry data from encoders on the wheels.
- Range-bearing measurements from the robot position.
- Correspondences between measurements and landmarks in the map c 0:t

And also the landmark-based map. Finally, it also contains the ground truth of the robot poses, which you should use to compare to your solutions.

4.1 The main GraphSLAM loop

The steps to initialize and iterate until convergence are detailed in the table below. The skeleton and GraphSLAM initialize() have been provided, you may need to choose a reasonable convergence threshold or maximum number of iterations at the top of the graph slam file. There is also a provision for breaking up longer datasets into sub-parts which you will find helpful.

The program is run from the graph slam script. That script after loading the data and initializing runs a loop over GraphSLAM linearize(,,,) and GraphSLAM reduce(...) followed by solving the resulting system. You should try the various data sets by uncommenting the simOutFile and mapFile names in pairs.

As you will find the data sets get increasingly difficult to make converge. You may want to try adjusting the parameters in the script and the Q and R matrices. By the third pair of files you will find that the default settings do not quite work.

4.2 Questions/exercises

8. Characterize each of the five data sets:
   - What properties make them easier or harder?
   - Do you see evidence of local minima or very weak minima?
   - How does density of the map and the sensor range effect the convergence?
   - How does the amount of noise effect results?
   - What about the starting point?
   - When is the linearize system a reasonable approximation or not?
   - What happens when the robot closes a loop returning to see a landmark from the beginning?
9. Save images of the best final map for each data set and include with your report.
10. Suggest some ways that the method could be improved, (for instance, how might we deal with outliers and spurious measurements, how to avoid oscillation and overshooting or what if data association, the feature indices of the measurements, were not known.).
11. (optional) Implement some (non-trivial) ways of improving the performance and show the improvement (or lack of). (2 points for each improvement up to 4 points maximum,)

5 References

[1] D. Koller and N. Friedman, Probabilistic graphical models: principles and techniques. MIT press, 2009.

[2] S. Thrun, W. Burgard, and D. Fox, Probabilistic robotics. MIT press, 2005.

[3] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31–43, 2010.

[4] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," The International Journal of Robotics Research, vol. 25, no. 12, pp. 1181–1203, 2006.

[5] T. B. Schön and F. Lindsten, "Manipulating the multivariate gaussian density," Division of Au- tomatic Control, Linköping University, Sweden, Tech. Rep, 201